



ثانوية التكنولوجيا التطبيقية
Applied Technology High School

Microcontrollers

Module 3: Digital Display



PREPARED BY

Academic Services Unit

April 2012

© Applied Technology High Schools, 2012

Module 3: Digital Display

Module Objectives

Upon successful completion of this module, students will be able to:

- 1) Describe the purpose of a 7-segment display.
- 2) Describe how to read a 7-segment display pin map.
- 3) Use 'DIRH' and 'OUTH' variables in the PBASIC program to control the BASIC Stamp I/O pins.
- 4) Use 'LOOKUP' and 'LOOKDOWN' PBASIC commands as a means for referencing the lists of values used to display letters and numbers.
- 5) Assemble and test a 7-segment display circuit that displays letters and numbers.
- 6) Assemble and test a 7-segment display circuit that displays the given word.

Module Contents:

| | Topic | Page No. |
|-----|--|----------|
| 3.1 | Introduction to 7-Segment Display | 2 |
| 3.2 | Programming with DIRH and OUTH Variables | 4 |
| 3.3 | Keeping Lists of On/Off Patterns | 7 |
| 3.4 | Lab Activity 1 | 9 |
| 3.5 | Lab Activity 2 | 12 |
| 3.6 | Lab Activity 3 | 15 |
| 3.7 | Lab Activity 4 | 17 |
| 3.8 | Review Exercise | 21 |
| 3.9 | Assignment | 24 |

3.1 Introduction to 7-Segment Display

3.1.1 The Everyday Digital Display

Figure 3.1 shows a display on the front of an oven door. When the oven is not in use, it displays the time. When the oven is in use, it displays the oven's timer, cooking settings, and it flashes ON and OFF, at the same time an alarm sounds to let you know the food is done. A microcontroller inside the oven door monitors the pushbuttons and updates the display. It also monitors sensors inside the oven and switches devices that turn the heating elements ON and OFF.



Figure 3.1: Digital Clock 7-segment Display on Oven Door

3.1.2 What is a 7-Segment Display?

Each of the three digits in figure 3.1 is called a 7-segment display. In this module, you will program the BASIC stamp to display numbers and letters on a 7-segment display.

A 7-segment display is a package with 7 LEDs arranged in a shape that is useful for displaying digits and some letters. Figure 3.2 shows a part drawing of the 7-segment display you will use in this module's lab activities. Each of the segments (A through G) contains an LED that can be controlled individually. There is also a dot with an LED that can be used as a decimal point. Most of the pins have a number along with a label that corresponds with one of the LED segments. Pin 5 is labeled DP, which stands for decimal point.

Pins 3 and 8 are labeled “common cathode”, and they will be explained when the schematic for this part is introduced.

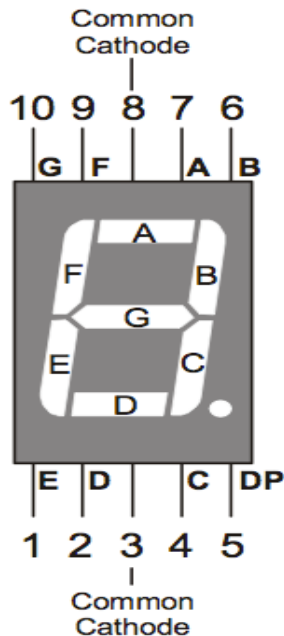


Figure 3.2A: 7-Segment LED Display Part Drawing and Pin Map

| Pin No | Mapping |
|--------|------------------------|
| 1 | Controls segment E |
| 2 | Controls segment D |
| 3,8 | Common Cathode |
| 4 | Controls segment C |
| 5 | Controls decimal point |
| 6 | Controls segment B |
| 7 | Controls segment A |
| 9 | Controls segment F |
| 10 | Controls segment G |

Figure 3.2B: 7-segment display Pin Mapping

Figure 3.2A shows a schematic of the LEDs inside the 7-segment LED display. Each LED anode is connected to an individual pin. All the cathodes share a common connection, therefore, the 7-segment LED display can be called a “common cathode” display. By connecting either pin 3 or pin 8 to Vss, you will connect all the LED cathodes to Vss.

The table in figure 3.2B shows the pin mapping, they help you connect the 7-segment display in your circuit by informing you of the pin number, pin name and pin reference.

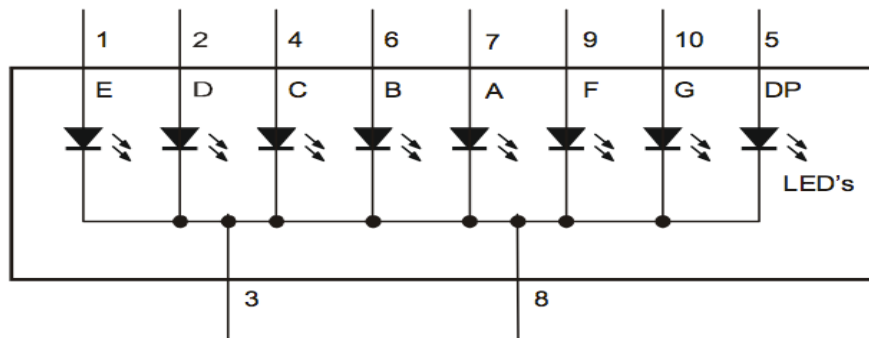


Figure 3.3: 7-Segment Schematic

Conduct Lab Activities 1 and 2.

3.2 Programming with DIRH and OUTH Variables

Including the decimal point there are eight different BASIC Stamp I/O pins that send high/low signals to the 7-segment display. That's eight different **HIGH** or **LOW** commands just to display one number. If you want to count from zero to nine that would be a huge amount of programming that will occupy a huge memory space. Fortunately, there are special variables you can use to set the high and low values for groups of I/O pins. By setting special variables called **DIRH** and **OUTH** equal to the binary numbers, you will be able to control the high/low signals sent by all the I/O pins connected to the 7-segment display circuit with a single PBASIC command.

DIRH is a variable that controls the **direction** (input or output) of I/O pins P8 through P15. **OUTH** controls the **high or low signals** that each I/O pin sends. "H" stands for high and controls pins P8-P15. It could be "L" (OUTL) for Low (P0-P7).

Figure 3.4 shows how you can use the DIRH and OUTH variables to control the direction and state (high/low) of I/O pins P8 through P15.

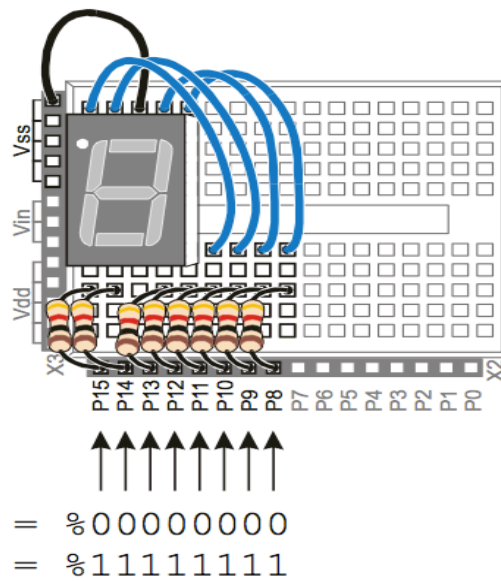


Figure 3.4: Using DIRH and OUTH commands to set all I/O pins to Output-low

The first command: `OUTH = %00000000` gets all of the I/O pins (P8 through P15) ready to send the low signals. If they all send low signals, it will turn all the LEDs in the 7-segment display off. If you wanted all the I/O pins to send the high signal, you could use **`OUTH = %11111111`** instead.

What does % do? The % is used to tell the BASIC Stamp Editor that the number is a binary number. For example, the binary number `%00001100` is the same as the decimal number 12.

The I/O pins will not actually send the low signals until you use the **DIRH** variable to change all the I/O pins from input to output. The command: `DIRH = %11111111` sets all I/O pins P8 through P15 to output. As soon as this command is executed, P8 through P15 all start sending the low signal. You can also use **`DIRH = %00000000`** to change all the I/O pins back to inputs.

All BASIC Stamp I/O pins start out as inputs. This is called a “default”. You have to tell a BASIC Stamp I/O pin to become an output before it starts sending a high or low signal. Both the **HIGH** and **LOW** commands automatically change a BASIC Stamp I/O pin’s direction to output. Placing a 1 in the DIRH variable also makes one of the I/O pins an output.

Figure 3.5 shows how to use the **OUTH** variable to selectively send high and low signals to P8 through P15. A binary-1 is used to send a high signal, and a binary-0 is used to send a low signal. This example displays the number “3” on the 7-segment display:

```
'      BAFG.CDE
OUTH = %11010110
```

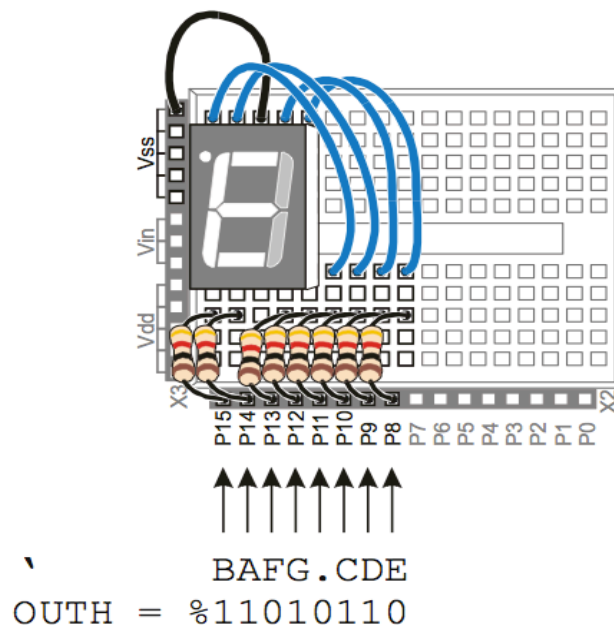


Figure 3.5: OUTH command demonstration

Inside the HIGH and LOW commands: The command HIGH 15 is really the same as OUT15 = 1 followed by DIR15 = 1.

Conduct Lab Activity 3

3.3 Keeping Lists of On/Off Patterns

LOOKUP Command: The **LOOKUP** command makes writing code for 7-segment patterns much easier. The **LOOKUP** command lets you “look up” elements in a list. It will help you select an item from the list, or compare an item to a list of items.

Command Syntax: LOOKUP index, [item1, item2,...], value

Here is a code example that uses the **LOOKUP** command:

LOOKUP index, [7, 85, 19, 167, 28], value

There are two variables used in this command, **index** and **value**. If the **index** is 0, **value** stores the **7**. If **index** is **1**, the **value** stores **85**.

```
' What's a Microcontroller - SimpleLookup.bs2
' Debug a value using an index and a lookup table.

' { $STAMP BS2}
' { $PBASIC 2.5}

value          VAR    Byte
index          VAR    Nib

index = 2

DEBUG ? index

LOOKUP index, [ 7, 85, 19, 167, 28], value

DEBUG ? value, CR
```

Since **index** is 2, in this example, the **LOOKUP** command places **19** into **value**, and that’s what the Debug Terminal displays.

LOOKDOWN Command: LOOKDOWN command compares a value with items in a list and informs the position (index) of the first item that fits the comparison criteria. It is the reverse of the LOOKUP command. When the LOOKUP command gives you a number based on the index, the

LOOKDOWN command gives you an index based on a number.

Command Syntax: LOOKDOWN value, comparison[item1, item2,...], index

The example program "SimpleLookdown.bs2" demonstrates how the LOOKDOWN command works.

```
' What's a Microcontroller - SimpleLookdown.bs2
' Debug an index using a value and a lookup table.

' { $STAMP BS2}
' { $PBASIC 2.5}

value          VAR      Byte
index          VAR      Nib

value = 167

DEBUG ? value

LOOKDOWN value, [ 7, 85, 19, 167, 28] , index

DEBUG ? index, CR
```

The value of index in this case would be "3".

Conduct Lab Activity 4.

3.4 Lab Activity 1

Objective: To manually build a circuit and test each segment in a 7-segment display.

Background: In the first part of this activity, you will manually build a circuit with 7-segment display and resistors, and ensure that each segment lights when power is applied to that particular segment. In the second part, you will display number '3' and the letter 'A' using the 7-segment LED circuit.

Equipment:

1. 7-segment LED display
2. Resistors – 1 k Ω (brown-black-red)
3. Jumper wires

Procedure: Part A

- With power disconnected from your Board of Education, build the circuit shown in Figure 3.6 and Figure 3.7.

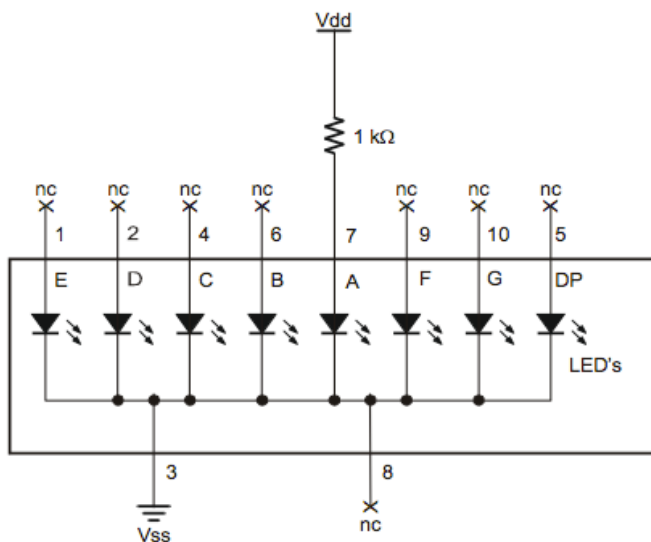


Figure 3.6: Test Circuit Schematic for Segment "A" LED display

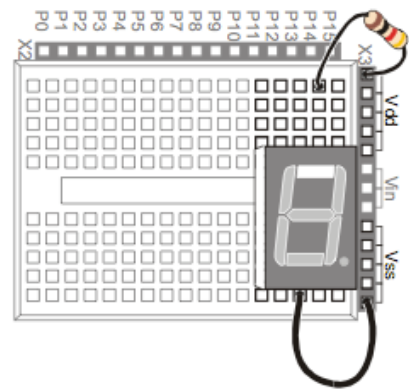


Figure 3.7: Test Circuit Wiring Diagram for Segment "A" LED display

- Reconnect power and verify that the A segment emits light.
- Disconnect power, and modify the circuit by connecting the resistor to the B LED input as shown in Figure 3.8 and Figure 3.9.

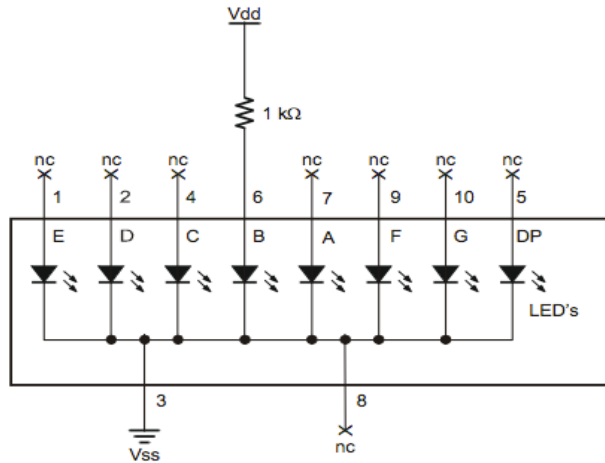


Figure 3.8: Test Circuit Schematic for Segment "B" LED display

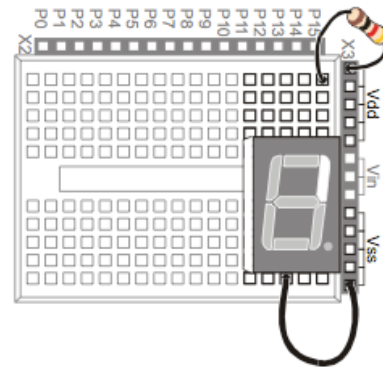


Figure 3.9: Test Circuit Wiring Diagram for Segment "B" LED display

- Reconnect power and verify that the B segment emits light. **Note:** Using the pin map from figure 3.2, this procedure could be adopted to test all other segments.

Procedure: Part B

- Build and test the circuit shown in Figure 3.10 and Figure 3.11, and verify that it displays the number three.

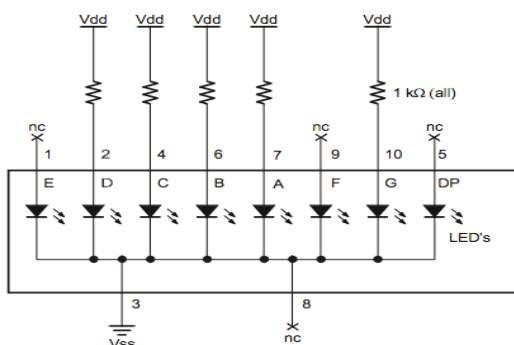


Figure 3.10: Test Circuit Schematic for digit "3" LED display

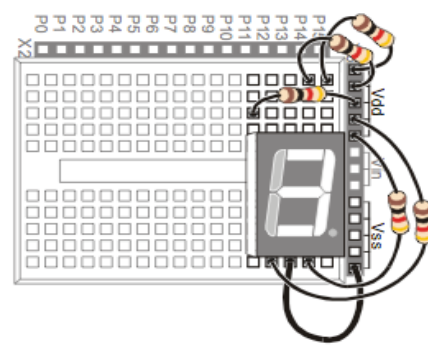
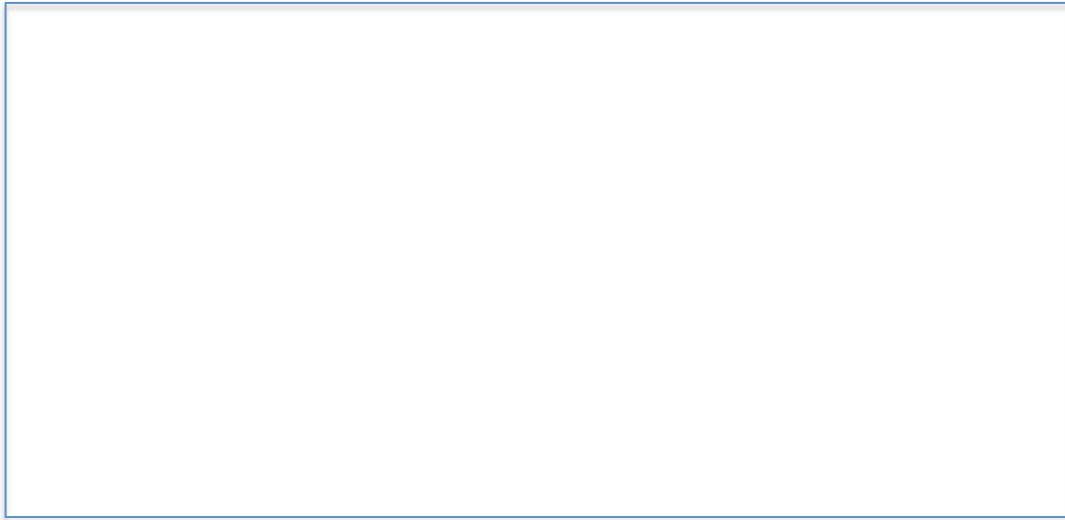


Figure 3.11: Test Circuit Wiring Diagram for digit "3" LED display

- Draw a schematic for the letter "A".



- Complete the Pin Map for the letter "A".

| Pin No | Mapping |
|--------|---------|
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

- Build and test the circuit to make sure it works. If letter "A" is not displayed, troubleshoot and identify the cause. Note down what went wrong.

3.5 Lab Activity 2

Objective: To build and control the 7-segment display LED circuit.

Background: In this activity, you will connect the 7-segment display to the BASIC stamp, and then run a simple program to test and make sure each LED is properly connected. The HIGH and LOW commands will accept a variable as a pin argument. Therefore, they are placed in a FOR...NEXT loop, and index is used to set the I/O pin, high, then low again.

Equipment:

1. 7-segment LED display
2. Resistors – 1 k Ω (brown-black-red)
3. Jumper wires

Procedure:

- Build the circuit shown in figure 3.12 and figure 3.13.

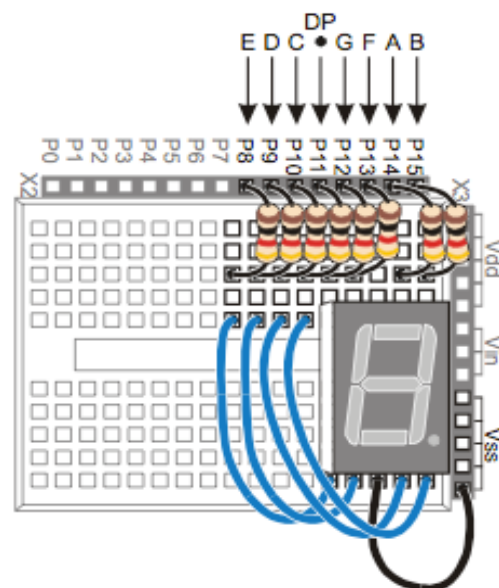
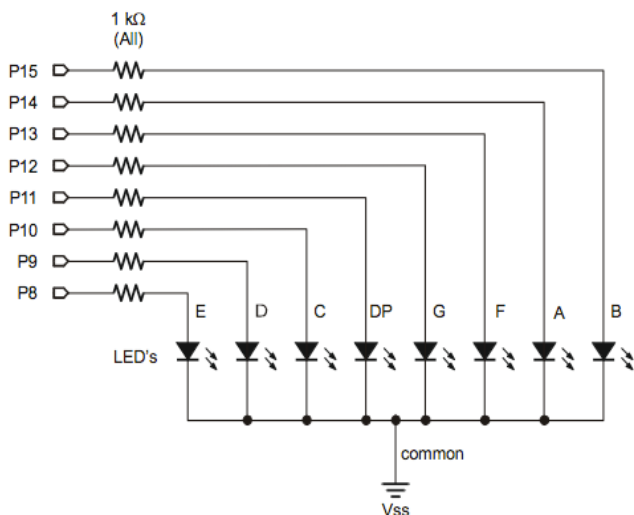


Figure 3.12: BASIC Stamp Controlled 7-Segment LED Display Schematic

Figure 3.13: Wiring Diagram for 3.12

- Program and the BASIC Stamp by following the steps below:
 - Enter and run SegmentTestWithHighLow.bs2.
 - Verify that every segment in the 7-segment LED displays lights briefly, turning on and then off again.
 - Record a list of which segment each I/O pin controls.

Example Program: SegmentTestWithHighLow

```
' What's a Microcontroller - SegmentTestWithHighLow.bs2
' Individually test each segment in a 7-Segment LED.

'{ $STAMP BS2

'{ $PBASIC 2.5

pinCounter      VAR      Nib

DEBUG "I/O Pin", CR,
      "-----", CR

FOR pinCounter = 8 TO 15

    DEBUG DEC2 pinCounter, CR
    HIGH pinCounter
    PAUSE 1000
    LOW pinCounter

NEXT
```

Conclusion:

What do you observe?

3.6 Lab Activity 3

Objective: To build the 7-segment display LED circuit and program it to cycle the 7-segment display through the digits 0 through 9.

Background: In this activity, you will experiment with the variables DIRH and OUTH. DIRH is a variable that controls the direction (input or output) of I/O pins P8 through P15. OUTH controls the high or low signals that each I/O pin sends. As you will soon see, OUTH is especially useful because you can use it to set the high/low signals at eight different I/O pins at once with just one command. Here is an example program that shows how these two variables can be used to count from 0 to 9 on the 7-segment display without using HIGH and LOW commands.

Equipment:

1. 7-segment LED display
2. Resistors – 1 k Ω (brown-black-red)
3. Jumper wires

LED Circuit:

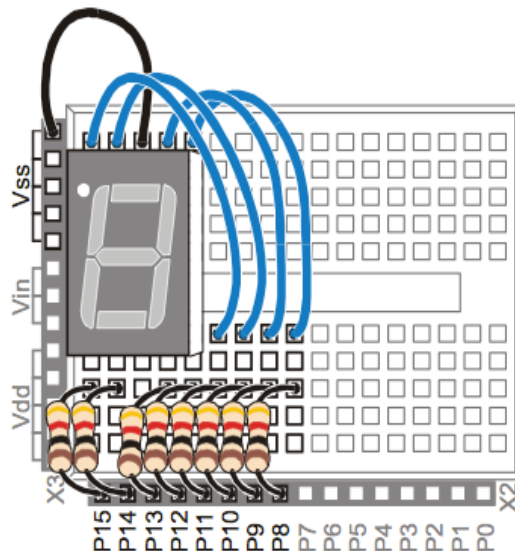


Figure 3.14: LED Circuit to display digits 0 to 9

Procedure:

- Build the circuit shown in figure 3.14.
- Enter and run DisplayDigits.bs2.
- Verify that the digits 0 through 9 are displayed.

Program: DisplayDigits.bs2

```
' What's a Microcontroller - DisplayDigits.bs2
' Display the digits 0 through 9 on a 7-segment display.

'{ $STAMP BS2}
'{ $PBASIC 2.5}

OUTH = %00000000      ' OUTH initialized to low.
DIRH = %11111111      ' Set P8-P15 to all output-low.
                        ' Digit:
'      BAFG.CDE
OUTH = %11100111      ' 0
PAUSE 1000
OUTH = %10000100      ' 1
PAUSE 1000
OUTH = %11010011      ' 2
PAUSE 1000
OUTH = %11010110      ' 3
PAUSE 1000
OUTH = %10110100      ' 4
PAUSE 1000
OUTH = %01110110      ' 5
PAUSE 1000
OUTH = %01110111      ' 6
PAUSE 1000
OUTH = %11000100      ' 7
PAUSE 1000
OUTH = %11110111      ' 8
PAUSE 1000
OUTH = %11110110      ' 9
PAUSE 1000

DIRH = %00000000      ' I/O pins to input,
                        ' segments off.

END
```

Conclusion:

What do you observe?

3.7 Lab Activity 4

Objective: To build the 7-segment display LED circuit and program it using LOOKUP and LOOKDOWN commands.

Background: In this activity, you will experiment with LOOKUP and LOOKDOWN commands as a means for referencing the lists of values used to display letters and numbers.

Part A: Using LOOKUP Command

Procedure:

- Load and run SimpleLookup.bs2.
- Run the program as-is, with the **index** variable set equal to 2.
- Try setting the **index** variable equal to numbers between 0 and 4.
- Re-run the program after each change to the **index** variable and note which value from the list gets placed in the **value** variable.

Program: SimpleLookup.bs2

```
' What's a Microcontroller - SimpleLookup.bs2
' Debug a value using an index and a lookup table.

' { $STAMP BS2}
' { $PBASIC 2.5}

value          VAR      Byte
index          VAR      Nib

index = 2

DEBUG ? index

LOOKUP index, [ 7, 85, 19, 167, 28], value

DEBUG ? value, CR
```

Observation:

Record the values displayed for each index variable in the table below:

| Index | Value |
|-------|-------|
| 2 | |
| 0 | |
| 1 | |
| 3 | |
| 4 | |

Part B: Using LOOKDOWN Command**Procedure:**

- Load and run SimpleLookdown.bs2.

```
' What's a Microcontroller - SimpleLookdown.bs2
' Debug an index using a value and a lookup table.

' { $STAMP BS2}
' { $PBASIC 2.5}

value          VAR      Byte
index          VAR      Nib

value = 167

DEBUG ? value

LOOKDOWN value, [ 7, 85, 19, 167, 28] , index

DEBUG ? index, CR
```

- Run the program as-is, with the **value** variable set equal to 167, and use the Debug Terminal to observe the value of **index**.
- Try setting the **value** variable equal to each of the other numbers listed by the **LOOKDOWN** command: 7, 85, 19, 28.

- Re-run the program after each change to the **value** variable and note which value from the list gets placed in the **index** variable.

Observation:

Record the values displayed for each index variable in the table below:

| Value | Index |
|--------------|--------------|
| 167 | |
| 7 | |
| 85 | |
| 19 | |
| 28 | |

Unless you tell it to make a different kind of comparison, the **LOOKDOWN** command checks to see if a value is equal to an entry in the list. You can also check to see if a value is greater than, less than or equal to, etc. For example, to search for an entry that is less than or equal to the value variable, use the **<=** operator just before the first bracket that starts the list.

- Modify SimpleLookdown.bs2 by substituting this value and LOOKDOWN statement:
value = 35
LOOKDOWN value, <= [7, 19, 28, 85, 167], index
- Experiment with different values and see if the index variable displays what you would expect.

Observation:

What is the “value” displayed after the LOOKDOWN command is modified with a **<=** sign?

Part C: Using LOOKUP Command to display digits 0 to 9

- Build the circuit shown below. You had built the same circuit for activity 3.

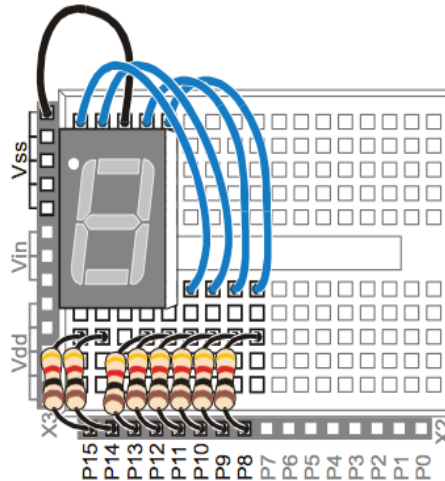


Figure 3.14: LED Circuit to display digits 0 to 9

- Enter the following program and verify that it displays digits 0 to 9.

```
What's a Microcontroller - DisplayDigitsWithLookup.bs2
' Use a lookup table to store and display digits with a 7-segment display.

'{ $STAMP BS2}
'{ $PBASIC 2.5)

Index          VAR      Nib

OUTH = %00000000
DIRH = %11111111

DEBUG "index  OUTH  ", CR,
      "-----  -----", CR

FOR index = 0 TO 9

  LOOKUP index, [ %11100111, %10000100, %11010011,
                  %11010110, %10110100, %01110110,
```

```
        %01110111, %11000100, %11110111, %11110110 ], OUTH
    DEBUG "    ", DEC2 index, "    ", BIN8 OUTH, CR
    PAUSE 1000
NEXT
DIRH = %00000000
END
```

- Observe the Debug terminal while the program runs to see how the value of index is used by the LOOKUP command to load the correct binary value from the list onto OUTH.
- Modify the program to display the letters of your first name. Complete the table given:

| Letter of your name | OUTH Variable |
|---------------------|---------------|
| | |
| | |
| | |
| | |
| | |
| | |

3.8 Review Exercise

1. In a 7-segment display, what is the active component that makes the display readable when the microcontroller sends a HIGH or LOW signal?

2. What does common cathode mean?

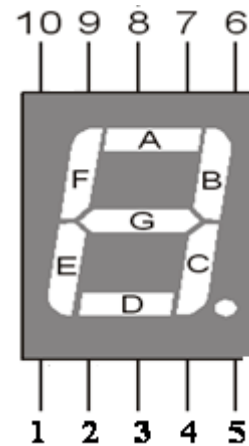
3. Write an OUTH command to set P8, P10, P12 high and P9, P11, P13 low. Assuming all your I/O pins started as inputs, write the DIRH command that will cause only the I/O pins mentioned as outputs.

4. Complete the table below by writing the values of OUTH required to display the letters given:

| Letter | LED Segments | BAFG.CDE | OUTH Value |
|--------|--------------|----------|------------|
| C | | | |
| d | | | |
| F | | | |
| S | | | |

5. To display digit '9' on the 7-segment display, identify the segments and pins that must be HIGH. Write your answers in the table below.

| Segment | Pin |
|---------|-----|
| | |
| | |
| | |
| | |
| | |
| | |



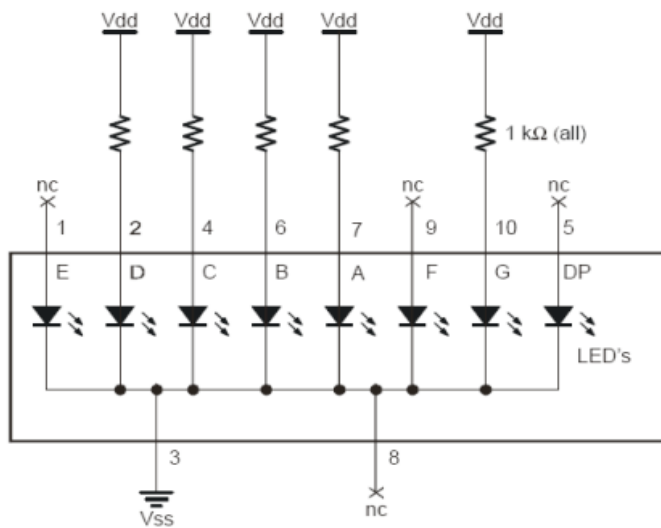
6. In the following command, if the Item = 75, what will be the value of Index?

LOOKDOWN Item, [11, 32, 15, 75, 243], Index

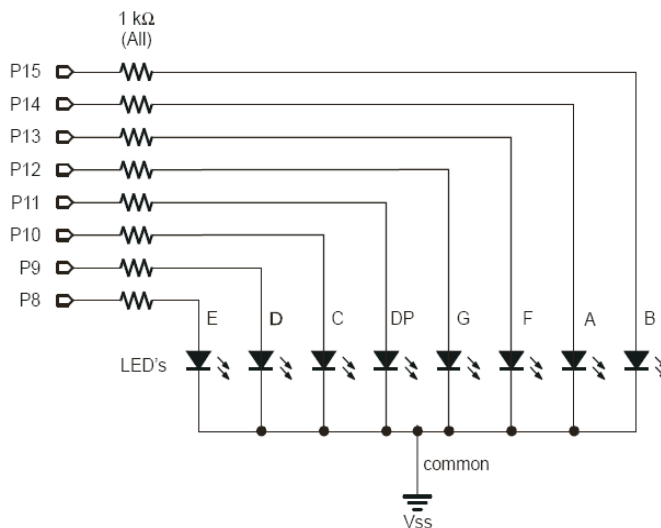
7. How many 7-segment displays are present in the display shown in the figure below.



8. The figure below shows the connection of a 7-segment display. What will be the number displayed?



9. The figure below shows the connection of a 7-segment display. Write a program below to display digit "9" for one second and then switch off the 7-segment display. Hint: Use DIRH and OUTH commands.



3.9 Assignment

Display "ATHS" over and over again on a 7-segment LED display. Make each letter last for a duration of 500 ms. Hint: Use 4 constants to store OUTH values of all letters, and use DO...LOOP and FOR...NEXT looping commands.